Name: *Student id:* *Section:* *Serial#:*

Question #	1	2	3	4	5	Total
Max points	12	20	8	16	20	76
Points earned						

University of Bahrain *College of Information Technology*
Department of Computer Science

ITCS242: Assembly Language Programming

Second test *Date: May 18, 2016* *Time: 90 minutes*
****

QUESTION ONE: Answer **any two** of the following questions as required.

1) Give no more than **6** instructions to store in a variable ***f*** ***sdword*** ***?;*** the product of multiplying the top two words stored on the stack. **{4 pts}**

```
pop    ax
pop    dx
imul   dx
mov    word ptr f, ax
mov    word ptr f+2, dx
```

2) Write the needed instructions to store in ***eax*** the count of non-digits in a string ***strX***.

strX ***byte*** ***"a,G;b9,56:tr,3#Q,7, F... "*** . **{8 pts}**

```
      mov     ecx, sizeof strx
      mov     eax, ecx
      mov     edx, 0;      counter
      mov     ebx, 0;      index
L1: cmp       strx[ebx], "0"
      jB      L2
      cmp     strx[ebx], "9"
      jA      L2
      inc     edx
L2: inc       ebx
      loop    L1
      sub     eax, edx
```

```
      mov     ecx, sizeof strx
      mov     eax, 0;  counter
      mov     ebx, 0;      index
L1: cmp       strx[ebx], "0"
      jB      L2
      cmp     strx[ebx], "9"
      jBE     L3
L2:  inc      eax
L3:  inc      ebx
      loop    L1
```

Name: *Student id:* *Section:* *Serial#:*

QUESTION TWO: Write a complete Assembly program that implements the following algorithm. **{20 pts}**

1) Define two variables ***sumE*** and ***sumD*** as required and initialize both of them to zero.
2) Data input:
   2.1. display a message that asks the user to enter from the keyboard one unsigned short value.
   2.2. enter from the keyboard unsigned short value in **hexadecimal**.
3) While (***the entered value*** is not equal to zero)
   3.1. if (***the entered value*** is even) then ***sumE = sumE + num;*** else ***sumD = sumD + num;***
   3.2. display a message that asks the user to enter from the keyboard the next unsigned short value.
   3.3. enter from the keyboard unsigned short value in **hexadecimal**.
   3.4. go to step 3.
4) At the beginning of a new line, display in decimal the values of ***sumE*** and ***sumD*** separated by tab.

```
          INCLUDE   IRVINE32.INC
          .DATA
M1        BYTE      "ENTER UNSIGNED short VALUE PLEASE: "
SUME      DWORD     0
SUMD      DWORD     0

          .CODE
MAIN      PROC
    WH:   LEA       EDX, M1
          CALL      WRITESTRING
          CALL      READHEX

          MOVZX     EDX, AX
          MOV       BL, 2
          CMP       AX, 0
          JE        DIS
          DIV       BL
          CMP       AH,0
          JE        EV
          ADD       SUMD, EDX
          JMP       WH
   EV:    ADD       SUME, EDX
          JMP       WH

   DIS:   CALL      CRLF
          MOV       EAX, SUME
          CALL      WRITEDEC
          MOV       AL, 9
          CALL      WRITECHAR
          MOV       EAX, SUMD
          CALL      WRITEDEC

          EXIT
MAIN      ENDP
          END       MAIN
```

Name: *Student id:* *Section:* *Serial#:*

QUESTION THREE:

What will be in the specified registers after executing each of the following codes? **{ 8 pts}**

a)
```
MOV   SP, 7C3AH
POP   BX
PUSH  EAX
```

SP = **7C 38** H

b)
```
MOV     AL, 0E0H
MOV     BH, 0ADH
IMUL    BH
```

AX = **0A 60** H

c)
```
MOV    AX, 2C4FH
MOV    BX, 0FFFEH
IMUL   BX
```

AX = **A7 62** H

d)
```
MOV  AX,3ACEH
MOV  BX,0FFFFH
CWD
IDIV BX
```

AX = **C5 32** H

e)
```
    MOV  AX,5A8FH
    CMP  AX,9000H
    JLE  LA
    SUB  AX, 9000H
    JMP  LB
LA: ADD  AX, 9000H
LB:
```

AX = **CA 8F** H

f)
```
    MOV  CX,7A5FH
    MOV  BX,87C4H
    ADD  BX, CX
    JNS  L1
    SUB  BX, CX
  L1:
```

BX = **02 23** H

g)
```
    MOV  AX,3A20H
    MOV  BX,1CFFH
LY: DEC  AL
    CMP  AL, BH
    JB   LY
```

AX = **3A 1F** H

h)
```
    MOV   AX,7CA4H
    MOV   ESP,768EH
    MOV   ECX,3
 L8: ADD   AX, CX
    PUSH  EAX
    LOOP  L8
```

SP = **76 82** H

Name: *Student id:* *Section:* *Serial#:*

QUESTION FOUR: Convert each of the following C++ codes into Assembly language.

1) *signed byte x ,y ;* ***// You have to properly define f only*** **{8 pts}**
*f = (x + y) * (x – y);*

```
f       sword   ?, ?

        movsx   ax, x
        movsx   bx, y
        add     ax, bx          ; ax = x + y
        mov     cx, x
        sub     cx, bx          ; cx = y – x
        imul    cx              ; dx: ax = (x + y) * (y – x)
        mov     f, ax
        mov     f+2, dx
```

2) **{8 pts}**

```
void funU (short a, short b, short &t)
{   t = a;
    if(a < b)
       t = b;
}
```

```
funU    proc    uses ebx ax bx, a: sword, b: sword, t: ptr sword)
        mov     ax, a
        mov     bx, b
        mov     ebx, t
        mov     [ebx], ax
        cmp     ax, bx
        jge     LU
        mov     [ebx], bx
  LU:   ret
funU endp
```

Name: *Student id:* *Section:* *Serial#:*

QUESTION FIVE: **{20 pts}**

Write a complete Assembly program that defines in the data segment four arrays ***Xar***, ***Yar***, ***qut***, and ***rem***, each consisting of 80 signed words. The program consists of two procedures described as follows:

a) A procedure named ***wdiv*** that accepts two signed words ***x*** and ***y,*** return the quotient ***Q*** and remainder ***R*** of dividing ***x*** by ***y***. *(Write the procedure **wdiv** in a form that allows using **invoke** statement).*

b) A procedure named ***main*** that fills arrays ***Xar*** and ***Yar*** by generating 80 random values in the range from −240 to +80, calls the procedure ***wdiv*** to store the quotient and remainder of dividing the corresponding elements of arrays ***arrX*** and ***arrY*** in the corresponding elements of arrays ***qut*** and ***rem***.

```
        include   irvine32.inc
        .data
xar     sword     80 dup(?)
yar     sword     80 dup(?)
qut     sword     80 dup(?)
rem     sword     80 dup(?)
        .code
;    ********************************************
wdiv   proc    uses esi edi ax,x:word,y:word, Q:ptr word, R:ptr word
       mov     esi, Q
       mov     edi, R
       mov     ax, x
       cwd
       idiv    y
       mov     [esi], ax
       mov     [edi], dx
       ret
wdiv       endp
;    ********************************************
main proc
     call      randomize
     mov       ecx, lengthof xar
     mov       ebx, 0

L1:  mov       eax, 321
     call      randomrange
     sub       eax, 240
     mov       xar[2*ebx], ax
     mov       eax, 321
     call      randomrange
     sub       eax, 240
     mov       yar[2*ebx], ax

     inc       ebx
     loop      L1

     mov       ecx, lengthof xar
     mov       ebx, 0
L2:  invoke    wdiv, xar[2*ebx], yar[2*ebx], addr qut[2*ebx], addr rem[2*ebx]
     inc       ebx
     loop      L2

     exit
main endp
     end       driver
```